AYVOS BİLGİ TEKNOLOJİLERİ

1.HAFTA ÖDEVLERİ

(Görüntü İşleme Nedir?)

(Python-OpenCv Kütüphanesi Nedir?)

(Nesne Tespiti Nasıl Çalışır?)

(YOLO Nesne Tespiti Modelleri ve Katmanları Nelerdir?)

(YOLO Versiyonları ve Farkları Nelerdir?)

Hazırlayan:

Burak POLAT

Stajyer

GÖRÜNTÜ İŞLEME NEDİR?

Görüntü işleme, dijital görüntülerin analiz edilmesi ve dönüştürülmesi işlemlerini içeren bir alandır. Bu süreç, bir görüntüdeki belirli özellikleri tespit etme, görüntüyü iyileştirme veya yeniden oluşturma amacı taşır. Görüntü işleme, genellikle bilgisayarla görü ve yapay zeka uygulamalarında kullanılır.

Temel olarak görüntü işleme şu adımları içerir:

1. **Görüntü Toplama:** Dijital bir kamera veya başka bir sensör aracılığıyla bir görüntü elde edilir.
2. **Görüntü İyileştirme:** Görüntünün daha net, keskin ve kullanılabilir olmasını sağlamak için filtreler, gürültü giderme, kontrast iyileştirme gibi işlemler yapılır.
3. **Görüntü Segmentasyonu:** Görüntü, anlamlı alt parçalara bölünür. Örneğin, nesnelerin veya yüzlerin tespiti için segmentasyon yapılabilir.
4. **Özellik Çıkartma:** Görüntüdeki önemli özellikler (kenarlar, renkler, şekiller vb.) çıkartılır. Bu adım, görüntülerin daha ileri düzeyde analiz edilmesi için gereklidir.
5. **Yorumlama ve Tanıma:** Görüntüdeki nesneler tanınır ve sınıflandırılır. Bu adımda, makine öğrenimi veya derin öğrenme teknikleri kullanılabilir.
6. **Sonuçları Görselleştirme veya Çıktı Üretme:** İşlenen görüntüler üzerinde yapılan değişiklikler görsel olarak sunulabilir veya belirli bir çıktı (örneğin, analiz raporu) elde edilebilir.

Görüntü işleme, tıbbi görüntüleme, güvenlik sistemleri, robotik, endüstriyel üretim, otonom araçlar gibi pek çok alanda kullanılır. Bu alanda **OpenCv,** MATLAB, ve TensorFlow gibi yazılım araçları yaygın olarak kullanılmaktadır.

Kısaca görüntü işleme, gelişmiş bir görüntü elde etmek veya ondan bazı yararlı bilgiler çıkarmak için bir görüntü üzerinde bazı işlemleri gerçekleştirme yöntemleridir.

OPENCV KÜTÜPHANESİ NEDİR?

OpenCv (Open Source Computer Vision Library), görüntü işleme ve bilgisayarla görü (computer vision) alanlarında en popüler açık kaynaklı yazılım kütüphanesidir. OpenCv, görüntülerin işlenmesi, analizi, tanımlanması ve görselleştirilmesi için geniş bir araç seti sunar. 2000 yılında Intel tarafından başlatılan bu kütüphane, özellikle hız ve verimlilik açısından oldukça popüler olmuştur. Hem masaüstü hem de mobil platformlarda kullanılabilir.

Temel özellikleri şunlardır:

1. **Görüntü İşleme ve Manipülasyon**: OpenCv, dijital görüntülerin işlenmesi için temel fonksiyonlar sunar. Bu, renk dönüşümleri, filtreleme, kenar algılama, morfolojik işlemler gibi işlemleri içerir.
2. **Görüntü Segmentasyonu**: Görüntüdeki nesnelerin veya bölgelerin belirli kurallara göre ayrılmasını sağlar. Örneğin, thresholding (eşikleme) ve bölgesel segmentasyon yöntemleri ile görüntü üzerinde analiz yapabilirsiniz.
3. **Özellik Çıkartma ve Tanıma**: Nesne tanıma ve görüntüdeki önemli özelliklerin çıkarılması (örneğin kenar tespiti, yüz tanıma, hareket algılama) için kullanılır. OpenCv, SIFT, SURF, ORB gibi algoritmaları içerir.
4. **Makine Öğrenimi Desteği**: OpenCv, makine öğrenimi ve derin öğrenme modellerini destekler. Kütüphane, çeşitli sınıflandırma algoritmaları (SVM, KNN, vb.) ve yapay sinir ağları (DNN modülü) ile entegrasyon sağlar.

5. **Gerçek Zamanlı Uygulamalar**: OpenCv, video işleme ve gerçek zamanlı uygulamalar için optimize edilmiştir. Görüntüler üzerinde hızlı analiz ve işlem yapmayı sağlar.
6. **Objeler ve Hareket Algılama**: OpenCv, video analizi, hareket algılama, nesne takibi ve yüz tanıma gibi gerçek zamanlı sistemlerde yaygın olarak kullanılır.
7. **3D Görüntü İşleme**: Derinlik haritaları ve stereo görüntüler ile 3D görüntü işleme özelliklerine sahiptir. OpenCv, stereo eşleşme, derinlik haritası oluşturma ve 3D rekonstrüksiyon konularında işlevler sunar.
8. **Kamera Kalibrasyonu ve Görüntü Düzeltme**: Kamera lensi bozulmalarını (distorsiyonları) düzeltmek için kullanılan algoritmalar sağlar. Ayrıca, tek ve çift kamera sistemi kullanarak stereoskopik görüntüler elde edebilir.
9. **Python Desteği**: OpenCv, Python dili ile geniş bir entegrasyona sahiptir. Python API'si üzerinden kolayca kullanılabilir ve bunun sayesinde makine öğrenimi ve derin öğrenme projelerinde popüler hale gelmiştir.

OpenCv kullanım alanları şunlardır:

1. **Yüz Tanıma**: OpenCv, yüz tanıma için hazır modeller ve algoritmalar sağlar. Bu teknoloji, güvenlik ve biyometrik kimlik doğrulama sistemlerinde yaygın olarak kullanılır.
2. **Nesne Takibi**: Hareket algılama ve nesne takibi için OpenCv'nin çeşitli algoritmalarından faydalanılabilir. Bu, güvenlik kameralarından veya otonom araçlardan gelen verilerin işlenmesi için kullanılır.
3. **Endüstriyel Görüntü İşleme**: Üretim hattında kalite kontrolü yapmak için görüntü işleme teknikleri kullanılır. Örneğin, ürünlerin yüzeyindeki hataları tespit etmek.
4. **Otonom Araçlar**: Otonom araçlar, çevrelerini analiz etmek için OpenCv kullanabilir. Görüntü işleme, yol işaretleri, trafik işaretleri ve engellerin tespiti için önemlidir.
5. **Tıbbi Görüntüleme**: MRI, CT taramaları veya X-ray gibi tıbbi görüntüler üzerinde analiz yapmak için OpenCv kullanılabilir. Örneğin, tümör tespiti veya organ segmentasyonu.
6. **Tarım ve Çiftçilik**: Tarım alanlarında, mahsul analizi, hastalık tespiti ve otomatik sulama sistemlerinde OpenCv kullanılabilir.

Kısaca, OpenCv, görüntü işleme ve bilgisayarla görü uygulamalarında oldukça güçlü ve kapsamlı bir kütüphanedir. Görüntüleri analiz etmek, işlemek ve anlamak için sayısız araç ve algoritma sunar. Hem araştırma hem de ticari projelerde yaygın olarak kullanılır. Python ile entegrasyonu sayesinde, özellikle derin öğrenme ve makine öğrenimi uygulamaları için oldukça tercih edilen bir araçtır.

NESNE TESPİTİ NASIL ÇALIŞIR?

Bir görüntüde veya videoda belirli nesnelerin konumlarını tespit etmeye yönelik bir bilgisayarla görü (computer vision) işlemidir. Nesne tespitinin temel amacı, bir görüntüdeki nesneleri tanımlamak ve onların sınırlarını (bounding box) çizmek, yani bu nesnelerin bulunduğu bölgeyi belirlemektir.

Nesne tespiti genellikle şu adımlarla çalışır:

1. **Öznitelik (Feature) Çıkartma:** Nesne tespiti, görüntüdeki anlamlı öznitelikleri çıkartarak başlar. Bu öznitelikler, nesnenin şekli, renk, doku, kenarları gibi özellikler olabilir. Öznitelik çıkarma, genellikle görüntü işleme tekniklerine dayanır ve çeşitli algoritmalar kullanılarak yapılır. Özellikler çıkartıldığında, bu öznitelikler, nesneleri tanımak için daha uygun hale gelir.

2. Özniteliklerin Tanımlanması ve Sınıflandırılması: Nesne tespiti için, çıkarılan özniteliklerin sınıflandırılması gerekir. Örneğin, bir görüntüde bir araba, insan veya hayvan gibi farklı nesnelerin tespiti yapılabilir. Bu sınıflandırma, genellikle makine öğrenimi veya derin öğrenme tekniklerine dayanır.

- Makine Öğrenimi Tabanlı Yöntemler
- Derin Öğrenme Tabanlı Yöntemler

3. Bölge Önerisi (Region Proposal): Nesne tespitinde, **region proposal** (bölge önerisi) adımı, bir görüntüde hangi bölgelerin nesne içerebileceği hakkında tahminlerde bulunur. Bu işlem, genellikle nesnelerin bulunduğu alanların doğru bir şekilde sınırlandırılmasına yardımcı olur.

- Selective Search
- RPN (Region Proposal Network)

4. Sınıflandırma ve Konumlandırma (Bounding Box): Bölge önerileri yapıldıktan sonra, her önerilen bölge bir sınıflandırma modeline gönderilir. Model, nesnenin türünü (örneğin, insan, araba) ve nesnenin tam konumunu belirler.

Bu adımda, her tespit edilen nesne için bir **bounding box** (sınırlayıcı kutu) çizilir. Bu kutu, nesnenin bulunduğu bölgeyi tanımlar. Bu aşamada, nesne tespitinin doğruluğunu artırmak için **Non-Maximum Suppression (NMS)** gibi teknikler kullanılır. NMS, aynı nesneye ait birden fazla öneriyi birleştirerek tek bir doğru kutu oluşturur.

5. Sonuçların Görselleştirilmesi: Son olarak, tespit edilen nesneler bir görüntü üzerinde çizilen sınırlayıcı kutularla gösterilir. Görüntüye, her nesnenin adı ve sınıflandırılması eklenebilir.

Nesne tespiti, görüntülerdeki nesneleri doğru ve hızlı bir şekilde tanımlama işlemi olup, birçok farklı algoritma ve tekniği içerir. Derin öğrenme yöntemleri, özellikle büyük veri setleri ve karmaşık nesneler için çok etkili sonuçlar sağlar. OpenCv gibi kütüphaneler, nesne tespiti işlemini kolaylaştıran araçlar sunar ve çeşitli yöntemlerle uygulanabilir.

YOLO NESNE TESPİTİ MODELLERİ VE KATMANLARI NELERDİR?

YOLO (You Only Look Once), gerçek zamanlı nesne tespiti yapan bir derin öğrenme modelidir. **YOLO'nun temel amacı, tüm görüntüyü tek bir geçişte analiz ederek nesneleri hızlı ve doğru bir şekilde tespit etmektir.** YOLO, özellikle hız ve doğruluk açısından tercih edilen bir modeldir ve nesne tespiti alanında büyük bir yenilik getirmiştir. Modelin çalışma prensibi, "tek bir ağ geçişiyle" nesnelerin sınıflarını ve konumlarını (bounding box) tespit etmektir.

YOLO'nun temel çalışma prensibi, görüntü üzerinde bir **grid (ızgara)** oluşturarak, her bir ızgara hücresinde bir nesne olup olmadığını belirlemek ve bu nesnenin sınıfını ve konumunu tahmin etmektir. Modelin ağ yapısı, genellikle **Convolutional Neural Network (CNN)** tabanlıdır ve çok sayıda katmandan oluşur. Bu katmanlar, görsel özellikleri çıkartarak, görüntüdeki nesneleri tespit etmeye yardımcı olur.

YOLO'nun temel yapısı ve katmanlarının özellikleri şunlardır:

1. Girdi Katmanı (Input Layer)

- YOLO modeline giren görüntü genellikle sabit bir boyutta (örneğin 416x416 veya 608x608) olmalıdır.
- Bu katman, görüntü verisini alır ve işlemeye başlar.
- Görüntü, bir dizi renk kanalından oluşur (RGB gibi)

2. Konvolüsyonel Katmanlar (Convolutional Layers)

- YOLO, görsel özellikleri çıkarmak için çok sayıda konvolüsyonel katman kullanır. Konvolüsyonel katmanlar, görüntüdeki kenarları, köşeleri ve desenleri öğrenir.
- Her bir konvolüsyonel katman, görüntüdeki farklı özellikleri öğrenmek için filtreler (kernels) kullanır. Bu katmanlar, daha önceki katmanlardan daha soyut ve yüksek seviyede özellikler çıkartır.
- Bu katmanlar, ReLU (Rectified Linear Unit) gibi aktivasyon fonksiyonları kullanarak doğrusal olmayan özellikleri öğrenir.

3. Max-Pooling Katmanları (Max-Pooling Layers)

- Max-pooling katmanları, görüntüdeki önemli özellikleri çıkarırken boyutları küçültmeye yarar. Bu katmanlar, görüntüdeki her bölge için en yüksek değeri seçer ve bu şekilde hesaplama gücünü azaltır.
- Max-pooling, konvolüsyonel katmanlardan sonra sıkça gelir.

4. Yapısal Katmanlar (Residual/Skip Connections)

- YOLOv3 ve sonrasındaki modellerde Residual Connections (atlamalı bağlantılar) kullanılır. Bu katmanlar, ağın daha derin olmasına rağmen eğitimdeki zorlukları aşmaya yardımcı olur. Bu yapı, daha hızlı öğrenme ve daha doğru sonuçlar elde edilmesini sağlar.

5. Çıkış Katmanı (Output Layer)

- YOLO'nun en önemli çıkış katmanı, her ızgara hücresinde tespit edilen nesnelerin sınıfını, bounding box koordinatlarını ve güven (confidence) skorlarını içerir.
- Bu katman, her hücre için belirli sayıda bounding box tahmin eder. Bu tahminler, nesnelerin konumlarını ve türlerini belirler.
- **Confidence Score**: Bir nesnenin gerçekten o hücrede olup olmadığını belirtir. Eğer bu skor düşükse, model o hücredeki nesneyi geçerli saymaz.
- **Bounding Box (X, Y, W, H)**: Nesnenin dikdörtgen sınırlarını belirler. X ve Y, kutunun merkez koordinatlarını; W ve H ise kutunun genişliğini ve yüksekliğini ifade eder.
- **Sınıf (Class)**: Görüntüdeki nesnenin hangi sınıfa ait olduğunu belirtir (örneğin, araba, insan, köpek, vb.).

6. Sınıf ve Konum Tahminleri

- YOLO, her bir ızgara hücresinde belli sayıda sınıf ve bounding box tahmin eder.
- Model, her bir tahmin için softmax fonksiyonu ile sınıf olasılıklarını hesaplar.
- Bounding box ve sınıf tahminleri yapıldıktan sonra, Non-Maximum Suppression (NMS) tekniği kullanılarak fazla tahminler birleştirilir. NMS, aynı nesneye ait birden fazla tahmin olduğunda, en yüksek güven skoruna sahip olanı seçer ve diğerlerini eleyerek sonuçları daha hassas hale getirir.

7. Çıktı Sonuçları:

- Bounding box koordinatları (X, Y, W, H) ile birlikte her nesnenin sınıfı ve güven skoru verilir.
- Nesne tespiti yapıldıktan sonra, model, her nesnenin etrafına bir kutu çizer ve her kutunun üzerinde nesnenin sınıf adı ile güven skoru yer alır.

YOLO, nesne tespiti alanında hız ve doğruluk açısından güçlü bir modeldir. Modelin katmanları, görüntüden özellik çıkarma, nesne sınıflandırma ve konumlandırma işlemleri ile nesneleri hızlı ve doğru bir şekilde tespit etmeye olanak tanır. YOLO'nun farklı versiyonları, modelin hızını, doğruluğunu ve verimliliğini artırarak, pek çok uygulama için en uygun çözüm sunmaktadır.

YOLO VERSİYONLARI VE FARKLARI NELERDİR?

1. **YOLOv7**, 2022 yılında yayımlanan ve YOLO (You Only Look Once) nesne tespiti serisinin önemli bir sürümüdür. YOLOv7, önceki sürümlerine kıyasla birkaç önemli iyileştirme ve yenilik sunarak daha yüksek doğruluk ve daha hızlı çıkarım süreleri sağlar.Bu sürüm, daha büyük modellerle daha fazla parametreye sahip olmasına rağmen, hızda önemli bir kayıp yaşamaz.

2. **YOLOv8**, YOLO'nun önceki versiyonlarına kıyasla önemli gelişmeler ve optimizasyonlar sunar. YOLOv8, özellikle hız, doğruluk ve model verimliliği konusunda iyileştirmeler yaparak daha etkili bir nesne tespiti çözümü sunmayı hedefler.Bu sürümde yapılan optimizasyonlar, gerçek zamanlı nesne tespitinde önemli bir avantaj sağlar.

3. **YOLO-NAS (You Only Look Once - Neural Architecture Search)**, YOLO serisinin bir çeşidi olup, **Sinirsel Mimari Arama (NAS)** teknolojisini kullanarak nesne tespiti için daha verimli bir model tasarımı sunar. YOLO-NAS, YOLO'nun hız ve doğruluk optimizasyonunu, NAS ile birleşiminde daha iyi bir hale getirir. NAS teknolojisi, özellikle daha hızlı çıkarım süreleri ve yüksek doğruluk oranları için modelin parametrelerini ve yapısını optimize eder.

4. **YOLO-World**, YOLO nesne tespiti modelinin dünya çapında genişletilmiş bir versiyonudur. Bu model, daha geniş bir kapsamda nesne tespiti yapabilmek için geliştirilmiş bir YOLO sürümüdür. "Dünya" kelimesi, bu modelin, çok çeşitli nesneleri tanıyabilme yeteneğini ifade eder. YOLO-World, temel olarak daha fazla kategoride nesne tespiti yapabilen ve çok daha geniş veri setleriyle eğitilmiş bir modeldir.

5. **YOLOv9,** bilgisayarla görme alanında önemli bir adım olarak öne çıkıyor. Daha yüksek doğruluk, geliştirilmiş eğitim süreçleri ve programlanabilir gradyan bilgisi kullanımı, bu sürümün daha verimli ve güçlü olmasını sağlıyor. Ayrıca, açık kaynaklı olması, araştırmacılar ve geliştiriciler için büyük bir fırsat sunuyor.

6. **YOLOv10**, nesne tespiti konusunda önemli bir adım atarak daha düşük parametrelerle daha yüksek hız ve doğruluk sağlar. Bu model, daha verimli bir nesne tespiti ve daha hızlı çıkarım süreleri sunarak, özellikle gerçek zamanlı uygulamalar ve düşük kaynaklı cihazlarda etkili bir seçenek olmuştur.

7. **YOLOv11**, YOLOv5 ve YOLOv8 modellerinin yaratıcısı olan Ultralytics tarafından geliştirilen bir bilgisayarlı görüş modeli mimarisidir. YOLOv11, nesne algılama, segmentasyon, sınıflandırma, anahtar nokta algılama ve yönlendirilmiş sınırlayıcı kutu (OBB) algılamayı destekler.

8. **YOLOv12**, dikkat mekanizmalarını etkin bir şekilde kullanarak nesne algılama alanında önemli iyileştirmeler sunar. Yüksek doğruluk ve hız gerektiren uygulamalar için güçlü bir seçenek olarak öne çıkmaktadır.

YOLO MODELLERİ BAŞARI VE HIZ KARŞILAŞTIRMALARI				
Model	***mAP (COCO)**	***FPS (T4 GPU)**	**Parametre Sayısı**	**Farkları**
YOLOv7	~45.2%	~90 FPS	35M	Gelişmiş dilate convolutions ve mosaic augmentation ile daha hızlı eğitim ve yüksek doğruluk.
YOLOv8	~47.2%	~85 FPS	40M	Değişken özellik çıkarımı ve yeni bir mimari yapı ile daha hızlı ve daha doğrusal çıkış.
YOLO-NAS	~48.1%	~75 FPS	25M	NAS (Neural Architecture Search) kullanarak daha az parametreyle yüksek doğruluk elde edilmiştir.
YOLO-WORLD	~48.5%	~70 FPS	35M	Dünya çapında büyük veri seti eğitimi ve özelleştirilmiş aktivasyon fonksiyonlarıyla daha iyi genel performans.
YOLOv9	~49.0%	~75 FPS	42M	Yeni optimizasyon teknikleri ve dikkat mekanizmaları (Attention Mechanism) ile daha hassas tespit.
YOLOv10	~50.2%	~95 FPS	45M	Flash Attention kullanımı ve düşük parametreli modeller ile hızda %25 azalma sağlanmış.
YOLOv11	~51.5%	~100 FPS	50M	Daha ileri dikkat mekanizmaları ve Transformer tabanlı katmanlar ile yüksek doğruluk ve hız.
YOLOv12	~52.0%	~105 FPS	55M	Çift-dikkat mekanizmaları, R-ELAN optimizasyonları ve daha derin model yapıları ile en yüksek doğruluk.

- ***mAP (COCO) :** Nesne algılama için Ortalama Hassasiyet.
- ***FPS (T4 GPU):** Bir görüntü işleme sisteminin saniyede kaç kare (frame) işleyebildiğini gösteren bir performans ölçüsüdür.

YOLO'nun (You Only Look Once) farklı versiyonları, nesne algılama konusunda önemli ilerlemeler kaydetmiştir. Her yeni sürüm, genellikle daha iyi doğruluk, daha yüksek hız ve daha verimli model yapılarına sahiptir. Bu sürümler arasındaki başarı ve hız farklarının sebepleri, kullanılan mimariler, katman yapıları, optimizasyon teknikleri ve eğitim stratejileriyle ilgilidir.

Hız Farklarının Sebepleri ve Katman Yapıları:

1. **Parametre Sayısı ve Model Karmaşıklığı**:

   YOLOv12 gibi daha yeni sürümler, daha derin model yapıları ve dikkat mekanizmaları kullanarak daha yüksek doğruluk sunar. Ancak, daha fazla parametre ve daha karmaşık yapılar, çıkış hızını bir miktar azaltabilir. **YOLOv10** ve **YOLOv11**, daha az parametre ile daha hızlı sonuçlar verirken, doğruluklarında çok büyük iyileştirmeler sunar.

2. **Dikkat Mekanizmaları (Attention Mechanism)**:

   YOLOv9 ve sonrasında kullanılan dikkat mekanizmaları (örneğin, Flash Attention, Transformer tabanlı yapılar), daha hızlı özellik çıkarımı ve nesne algılama sağlar. Bu katmanlar, modelin nesneye daha fazla odaklanmasını ve önemsiz bilgileri daha hızlı elemesini sağlar.

3. **Dilate Convolutions ve Mosaik Augmentation**:

   YOLOv7, daha hızlı eğitim süreci sağlamak için dilate convolutions kullanırken, aynı zamanda mosaic augmentation ile veri çeşitliliğini artırarak doğruluğunu iyileştirir. Bu teknikler, modelin daha geniş bir özellik yelpazesine dayalı olarak çalışmasını sağlar.

4. **Neural Architecture Search (NAS)**:

   YOLO-NAS, **Neural Architecture Search** teknolojisini kullanarak daha verimli ve küçük modeller geliştirmeyi başarmıştır. Bu sayede parametre sayısı azalır, fakat yine de yüksek doğruluk sağlanır.

5. **Transformer Katmanları**:

   YOLOv11 ve **YOLOv12**, daha ileri düzeyde Transformer tabanlı katmanlar kullanarak daha yüksek doğruluk sunmaktadır. Bu tür katmanlar, görsel bilgiyi daha iyi işleyerek hem hız hem de doğruluk açısından büyük fayda sağlar.

6. **Optimizasyon Teknikleri**:

   YOLOv10 ve sonrasında kullanılan R-ELAN (Residual ELAN) ve Flash Attention gibi optimizasyon teknikleri, modelin hesaplama gereksinimlerini düşürür ve çıkış hızını artırır. Bu optimizasyonlar, nesne algılama ve çıkış hızını artırırken, doğruluğu da iyileştirir.

Sonuç:

- **YOLOv7** ve **YOLOv8**, daha hızlı çıkarım yapabilen ancak biraz daha düşük doğruluk sağlayan modellerdir.
- **YOLOv9** ve **YOLOv10** arasında hız ve doğruluk arasında iyi bir denge bulunur, özellikle **YOLOv10** yüksek hızda çalışırken daha düşük parametre ile iyi bir performans gösterir.
- **YOLOv11** ve **YOLOv12**, hız ve doğruluk açısından en verimli modellerdir. Bu modeller, dikkat mekanizmaları ve Transformer tabanlı yapılar sayesinde yüksek doğruluk sunarken, performans kaybı minimum düzeydedir.